(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2001 (29.11.2001)

PCT

(10) International Publication Number
WO 01/90933 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/EP01/05869

(22) International Filing Date: 22 May 2001 (22.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
00650056.5 24 May 2000 (24.05.2000) EP

(71) Applicant *(for all designated States except US)*: **OPENWAVE SYSTEMS INC.** [US/US]; 800 Chesapeake Drive, Redwood City, CA 94063 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **HODGSON, Kevin** [GB/GB]; c/o Phone.com (Europe) Limited, Vo-Tec Centre, Hambridge Lane, Newbury, Bershire RG14 STN (GB).

(74) Agents: **WELDON, Michael, J.** et al.; c/o John A. O'Brien & Associates, Duncairn House, 3rd Floor, 14 Carysfort Avenue, Blackrock, County Dublin (IE).

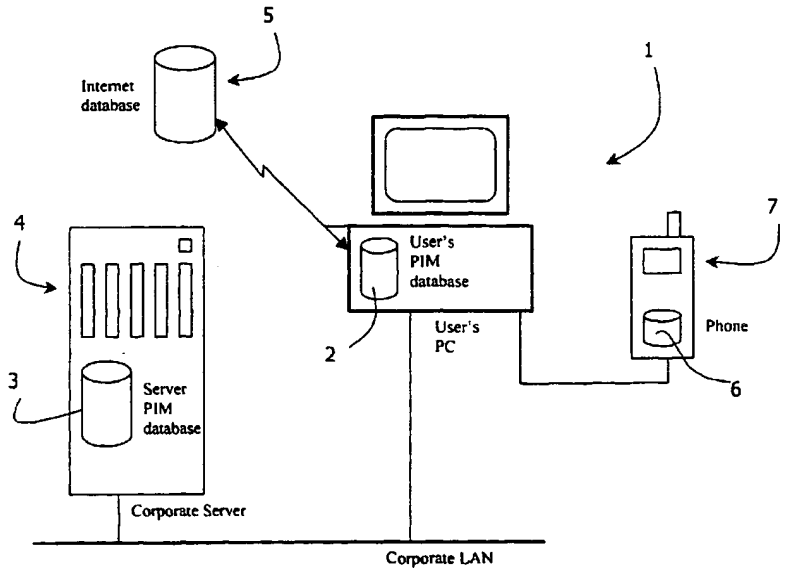
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DE (utility model), DK, DK (utility model), DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: SYNCHRONISATION OF DATABASES



(57) Abstract: Client databases are synchronised by storing a synchronisation engine (10) maintaining a synchronisation table (20) of record identifiers and change indicators for records of the client databases. The table (20) is used during a synchronisation session and is maintained persistent between sessions. All client databases (DBA-DBD) are synchronised in a session. Values for identifiers and change indicators are received from the client databases and are compared with the stored values to determine if a record has changed. The current value is thus identified. All of the client databases are updated with current value of a record and updating the synchronisation table is updated for a next synchronisation session.

WO 01/90933 A1

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

"Synchronisation of databases"

INTRODUCTION

Field of the Invention

The invention relates to synchronisation of data across multiple databases even if the databases have different structures. For example, a contact record in a mobile phone database should contain the same data as a record in an email system contact database of the user and referring to the same contact.

Prior Art Discussion

Heretofore, synchronisation is often performed by direct point-to-point updating for two databases. Where more than two databases are involved, synchronisation is typically achieved by daisy-chaining point-to-point sessions between pairs of databases. This approach, however, suffers from the problems of:-

(a) Synchronisation is not guaranteed to be complete since there is always at least one database that is not present in the process. It can be shown that three or more databases can always be in an unsynchronised state irrespective of how many times the links in the daisy chain are reapplied. This failure can occur even when the data has not been changed by the user; the changes being a result of the synchronisation itself.

(b) Complexity. As the number of clients grows, the number of possible point-to-point links increases. For example, five databases have ten different point-to-point links. Also, the process involves applying and reapplying links in order to ensure that all changes have been propagated within the system.

United States Patent Specification No. US5884325 (Oracle) describes synchronising shared data between computers having similar database structures. When a connection is established updates are propagated from the client to the server and vice versa. The server eventually propagates updates to other clients. Because this arrangement provides the server at the centre of a hub linked to all clients it appears to be an improvement over the direct point-to-point method described above. However, in this arrangement clients will often be out of synchronisation with each other. This has the same drawbacks as other point-to-point synchronisation methods.

The invention is therefore directed towards providing an improved synchronisation system and method.

SUMMARY OF THE INVENTION

According to the invention, there is provided a synchronisation system comprising a synchronisation engine comprising means for communication with a plurality of client databases and for updating the client databases with current data received by a client database, characterised in that,

the synchronisation engine comprises means for performing a synchronisation session in which all client databases (DBA-DBD) are updated with current data.

Thus, all client databases are synchronised at the end of the session until the next session. This is a major benefit for data integrity.

In one embodiment, said engine comprises means for maintaining a synchronisation table storing an identifier and a change indicator for each record of each client database, for comparing the stored change indicators with those read from the client

databases during a synchronisation session, and for maintaining the table in a persistent manner between synchronisation sessions.

In another embodiment, the system comprises a client interface associated with each client database, each client interface comprising means for providing a universal format for transfer of data to and from the engine independently of the client database format.

In one embodiment, the synchronisation engine comprises means for prompting user input of instructions if a conflict arises in which a record has changed in more than one client database.

In another embodiment, the engine comprises means for determining that there is a conflict if there are differences between records at the field level.

In a further embodiment, the engine comprises means for outputting representations of the conflicting fields in a universal format.

In one embodiment, the engine comprises means for adding or deleting databases as clients by adding or deleting corresponding datasets such as columns in the synchronisation table.

In another embodiment, the engine comprises means for maintaining a master database of current data.

In a further embodiment, the engine comprises means for using the master database as a client database when a client database is temporarily unavailable.

In one embodiment, the engine comprises means for temporarily masking a dataset for a temporarily unavailable client database.

In another embodiment, the engine comprises means for maintaining a status indicator in the synchronisation table to indicate availability status of a corresponding client database.

In one embodiment, the status values include uncreated, unsynchronised, synchronised, and null.

In another embodiment, a client interface comprises means for maintaining an intermediate database of true values for truncated values in the associated client database.

In a further embodiment, a client interface comprises means for maintaining a persistent database to allow transformation to provide a universal format to the engine.

In one embodiment, a client interface comprises means for maintaining a personalisation table containing a master identifier, a native identifier, an updated flag for every associated record, and a current record identifier list, and the client interface comprises means for using said data to maintain a client database having a subset of the records of other client databases.

According to another aspect, the invention provides a method for synchronising a plurality of client databases in a single synchronisation session, the method comprising the steps of:

storing a synchronisation table of record identifiers and change indicators for records of the client databases;

receiving the values for said identifiers and change indicators from the client databases, and comparing the received values with the stored values to determine if a record has changed to determine the current value; and

updating all of the client databases with a current value of a record and updating the synchronisation table for a next synchronisation session.

DETAILED DESCRIPTION OF THE INVENTION

Brief Description of the Drawings

The invention will be more clearly understood from the following description of some embodiments thereof given by way of example only with reference to the accompanying drawings in which:-

Fig. 1 is a schematic representation of a number of databases interconnected for synchronisation,

Fig. 2 is a generalised diagram illustrating the architecture;

Fig. 3 is a schematic representation of a synchronisation table generated during a synchronisation process;

Figs. 4(a) and 4(b) are together a flow diagram of synchronisation;

Figs. 5 and 6 are sample tables indicating system states;

Fig. 7 is a sample conflict resolution dialog;

Fig. 8 is a set of sample records and corresponding entries in a synchronisation table;

Fig. 9 is a representation of sync. table columns to handle disconnected clients;

Fig. 10 is a representation of handling of truncated fields;

Fig. 11 is a logical view of a transformation operation; and

Fig. 12 is a representation of personalisation components.

Description of the Embodiments

Referring to Fig. 1 a user's PC 1 is programmed to perform a process to synchronise a database 2 on the PC 1, a database 3 in a server 4, an Internet database 5, and a database 6 of a mobile phone 7. This is one example scenario. At a general level and referring to Fig. 2, synchronisation is performed by a synchronisation system comprising:

a synchronisation engine 10, and

a client interface ("client") 11 associated with each application A, B, C, and D having an associated API and a database DBA, DBB, DBC, and DBD respectively.

For synchronisation there are multiple databases and so the basic architecture is in the form of a hub with the engine 10 in the centre and a client 11 for each database.

The engine 10 generates a synchronisation table 20, as shown in Fig. 3. This table has a row for each item (such as a contact) and two columns for each database. The two columns contain for each item a unique identifier (UID) and a change indicator (CI). These two columns are derived directly from the databases. The CI is the "last modified timestamp" where available, and if not available a Cyclic Redundancy Check (CRC) may be generated directly from the data having a high probability of detecting that a record has changed.

When synchronising a record across the databases the UID and CI for each of the corresponding records from each of the databases is stored. They are stored in a logically linked manner, so that in future synchronisation sessions the linked records can be compared 'like-with-like'. The synchronisation table contains the UIDs and the CIs for all synchronised records in all of the configured client databases (synchronisation points). Each column in the synchronisation table corresponds to all of the synchronised records for one of the databases. Each row in the synchronisation table corresponds to a record that is linked by synchronisation between data stores. The synchronisation table is persistent, i.e. its state is saved between synchronisation sessions.

The engine 10 is an independent engine that provides all of the synchronisation processing, including duplicate handling, conflict resolution, and overall control of the synchronisation process. It invokes the individual clients 11, and requests from them data about the current state of their devices/applications and instructs them to write back information to the device/applications. It provides a generalised interface for each different device or application. The engine 10 is responsible for maintaining the synchronisation data.

Each client 11 is aware of the format and API offered by the associated application and handles all communication with it. Each client 11 isolates the engine 10 from

the specific detail of how information is stored in the applications. The clients 11 also provide:

- Application-specific configuration dialogs.
- Application-dependent storage of non-synchronisable data, such as voice dialling tags on phones. This information is managed by the client 11.

The APIs are conventional APIs for third party access to content. The synchronisation process uses these APIs to read and write information to the databases in their own proprietary formats.

The interface between the engine 10 and the clients 11 defines a common and consistent universal format containing all of the fields required for synchronisation. The engine 10 stores a master database of all synchronised data in the universal format.

Referring to Figs. 4(a) and 4(b), the synchronisation process uses the CI value from the clients 11 to determine (by checking with the synchronisation table) if there is an edit conflict. Resolving the conflict involves writing to the universal (or master) database the value from the database having the most recent update as indicated by the CI. If more than one database has an updated CI then user input may be required. Where a new record is detected the engine 10 creates a new row in the sync. table. If the "new" record is a duplicate the engine 10 makes a link to an existing record after searching key fields and possibly with user input. Any conflict with the new record is resolved as described above, again possibly with user input.

When the engine 10 has processed all records from all client databases it has an updated master database. It then updates the client databases by deleting required records, updating edited records, creating new records and applying personalisation (described in more detail below). The sync. table is then saved to file.

In the above process, there is conflict resolution where a record representing the same information has been amended in more than one client database. This is known as a conflict and the records are termed as conflicting records. In the case of a conflict, there is a need to enable the user make a decision as to which record accurately reflects the current data and communicate that decision to the synchronisation engine. Furthermore, there may be occasions where a number of fields within each record have been modified where the current correct data is in fact spread across the conflicting records. In this latter case a simple selection of one of the conflicting records as the master would not ensure that the correct information is written back to each of the client databases as a result of the synchronisation process. It is of significant benefit for the user to be able to select individual fields from the records in conflict to produce a merged new record, so that all fields in the synchronised record are up-to-date at the end of synchronisation.

A conflict can also occur if there is a new record in one of the applications, which is being synchronised for the first time, that has been identified to be duplicate of one that already exists in the synchronised record set (by using the key fields to identify the duplicate). In this case, the non-key fields (i.e. those which were not used to identify a new contact as an existing one) are candidates for conflict resolution.

During the synchronisation process, the engine 10 retrieves all of the edited records and new records from all the applications in the universal record format via the clients 11. Having all records available in the universal format allows field-level conflicts to be detected consistently:

Where a synchronised record has been edited in more than one application, the edited universal record representations are compared field-by-field to determine which fields of the record are in conflict. If there are fields in conflict, the conflicting fields are identified to the user, via a user interface so that they are able to identify the correct data, in other words, resolve the conflict. If no conflicting fields are

found, because each of the changed fields in each of the conflicting records were changed to the same new value(s), then there is no conflict to resolve even though the records are in conflict. In this latter case, the solution is intelligent enough to proceed without the need for user intervention.

Fig. 5 illustrates the state of the system after the last synchronisation and Fig. 6 illustrates the state of the system prior to the next synchronisation when the same record has been edited in both applications. The column marked K is the user's key field for the record (i.e. what they would typically use to identify the record; in the case of a contact database this might be the contact's full name). The synchronisation table indicates that the two records in the separate applications represent the same information. Comparison of the change indicators (CI) for each application in the row of the synchronisation table with the values stored in the application indicate that the record has changed in both applications. Field by field comparison indicates that there are two fields in conflict, namely the first and the forth fields. It is important to note that, though the third field has changed (to Kiwis), it is not in conflict as the identical change has been made in both records. It should also be noted that the forth field is in conflict even though a change has been made to that field in only one of the records.

In order for the user to be able to resolve the conflicting fields of the records, the user must be presented with the information in universal record format from each of the conflicting records, which have been collated during the conflict-detection phase.

The user interface for the conflict resolution consists of a PC-based dialogue displaying a table view whose columns correspond to the fields of the universal record. The header of the table identifies the content of each column. The first column of the table identifies for each row of the table where the rows data is derived from. The top row of the table view displays the resultant resolved record. This

record does not directly correspond to any real record in a database; it is the 'working' record, which the user is resolving.

The rows in the conflict resolution table view of Fig. 7 show the universal record representation from each of the applications for the conflicting records. There must always be at least two additional rows displayed, since these rows correspond to the conflicting records. To assist the user in the conflict resolution process, the columns of the table view are re-ordered, so that the fields that are in conflict are displayed on the far left of the table. In addition, the conflicting fields are highlighted in bold to distinguish them from the non-conflicting fields. This quickly draws the attention of the user to the fields which are in conflict, so that they may resolved with the minimum difficulty. The user chooses the most up-to-date fields from the conflicting options, by clicking directly on the field value required in the table view. This causes the field value in the resolved record (displayed in the top row) to be updated to this last selected value.

When the user has completed merging the conflicting records, he positively affirms the merge (by clicking on an OK button). This causes the fields currently displayed in the top row to be stored in the master database as the 'resolved record', and subsequently to be written by each of the clients to their respective applications.

There is an arrow head to the left of each of the rows representing conflicting records. This is a row selector button enabling the user to select a whole row as the row to be used.

The above describes the case where the same field is updated in all conflicting records to exactly the same value and this means that there is in fact no conflict to be resolved. There is one further case where there is no conflict to be resolved, even when a conflicting record is detected (i.e. the same record in multiple applications has been edited). This case occurs when the different applications support a different

subset of fields of the universal record. Fig. 8 illustrates this. In Fig. 8 we see that the field change in Application A' does not appear in Application B', and similarly the field change in Application B' does not occur in application A'. In this instance, a single merged universal record can be derived from the information in both the conflicting records without the need for user intervention.

Thus, the invention provides the following features:

Full field-level conflict resolution, with the ability to merge up-to-date fields from record edits in different data sources.

Ability to detect and handle the situation where identical changes have been made to the conflicting records and merge them without the need for user interaction.

Ability to detect and handle situations where changes have been made to conflicting records containing non-intersecting sets of changed fields. Again, in this instance, there would be no need for user interaction.

Addition and removal of client databases is achieved in a simple manner without destabilisation of the system.

When a new client is configured, a new column is added to the sync. table. The entries of the new column are initialised to have a synchronisation state of 'unsynchronised'. During the next synchronisation, this 'unsynchronised' state triggers the engine 10 to synchronise all the entries in the master database with the newly configured data store. When a client is removed, a column is removed from the table. This has no further effect on the synchronisation process, apart from reducing the number of client databases to be synchronised by one.

The invention also handles the situation where one of the clients is temporarily unavailable. This might occur, for example, if one of the databases to be synchronised happens to be a handheld device for which the battery has expired, or if the connection to the database is unreliable, as it might be for an Internet-based database. In such a case, it would be highly inconvenient if the unavailability of one of the configured databases were to prevent the synchronisation of the remaining databases. Provision of the synchronisation table together with the master database allows for clients to be temporarily unavailable during any given synchronisation. By masking out one of the columns of the table, thus hiding it from the synchronisation process, synchronisation of all the remaining available databases can continue. The existence of the master database ensures that the overall record-set is always available in universal format. When the databases becomes available again for synchronisation, its column of the table becomes visible to the engine, and its data set is synchronised.

To support disconnectable clients, the synchronisation table also contains a status field in the table, which is used by the engine 10 to determine the required action to take, as shown in Fig. 9. The status table entry may take on one of the following values: UNCREATED, UNSYNCED, SYNCED, or NULL.

If an attempt is made to create a record in a client 11 which is disconnected, the create fails, and status is set to UNCREATED. This records the fact that it failed, so that on subsequent synchronisation sessions, the create can be re-tried. If an attempt is made to edit a record in a client 11 which is disconnected, the edit fails, and the status is set to UNSYNCED. This records the fact that it failed, so that on subsequent synchronisations, the edit can be re-tried. If the client 11 is connected, so that creates/edits succeed, the status is set to SYNCED.

If a record is deleted in another (connected) client 11, the engine logic decides to propagate the delete across all clients 11. For all successful deletes, the

corresponding synchronisation table entry status is set to NULL. If one of the clients 11 is disconnected at the point the record delete request is made, the delete fails, and the status is set to UNSYNCED. Because the synchronisation table row is not removed from the table until all entries' status is NULL (indicating successful deletions), the engine is able to re-try the deletion.

The NULL status is also used as part of a personalisation solution; if a table entry's status for an external client 11 is NULL, and the status for the corresponding entry in the column for the master database is non-NULL, then the entry for the external client 11 represents a record that is excluded from the personalisation set.

This technique for handling disconnected clients is equally applicable to the case where the database is physically unavailable, or where the user has elected to omit a given database from the synchronisation. When configuring clients 11, there are three options available: (a) always synchronise the database, (b) prompt the user for whether to synchronise the database, or (c) never synchronise the database.

Thus, the invention provides the ability for the user to control, at synchronisation time, the database (clients) that will be synchronised. It also provides the ability to complete simultaneous multi-point synchronisation across a set of applications when one or more are not available to participate in the synchronisation. On the next time an application re-joins the synchronisation process, to be able to pickup "pending changes" from previous synchronisation sessions.

In the multipoint synchronisation process the aim is to ensure each participating database maintains as faithful a copy of the information stored in each other database as possible. In the situation where the structures of the different datastores differ, this may involve some transformation in the format of the data. There may also be limits on the faithfulness of the data representation imposed by the size limitations of the storage structures in the different datastores. In most cases when

synchronising information in PC, Server or Internet based databases the size limitation imposed by different databases do not practically affect the synchronisation as the fields are usually sized to cater for the majority of data that a user might use. Even if there is some truncation this usually occurs in a sufficiently small amount of cases to have negligible affect.

However, when providing multipoint synchronisation for mobile devices, a very real problem is faced. A typical mobile phone may limit the storage of a name in its contact datastore to say 12 characters. Synchronising a PC database such as Outlook™ to phone and back again presents a potential problem as illustrated below:

Original record in Outlook™

Key	Name	Number
A	Samantha Fielding	1-555-457-7845

Resulting record in phone (12 character name restriction)

Key	Name	Number
A	Samantha Fie	1-555-457-7845

Now, if a user changes the record on the phone, amending the number to 1-555-457-*8888,* the phone's record would read:

New record in phone

Key	Name	Number
A	Samantha Fie	1-555-457-8888

Using standard record-level synchronisation, the Outlook™ record would then become

Key	Name	Number

A	Samantha Fie	1-555-457-8888

This reduces the Outlook™ record to the storage limitation of the phone. Without special handling all datastores involved in the synchronisation risk being limited to the storage capabilities of the phone (for those fields stored on the phone). Referring to Fig. 10, in order to ensure correct synchronisation a client 11 maintains a persistent Intermediate Datastore to store a copy of the original universal record (or at least a copy of all the fields that could be subject to truncation) and its corresponding truncated data. This enables the client to detect changes to the truncated data in the application datastore by comparing the truncated data currently in the application datastore with the truncated data that was stored in the Intermediate Datastore.

The synchronisation is driven by the engine 10 requesting clients to read and write records to the databases. In a write request the clients are responsible for translating the record in universal record format as given by the engine 10 into the application's record format and conversely, in a read request the clients 11 are responsible for translating the record from the applications format to the universal record format. To handle truncation, the following additional processing is required (referring to Fig. 10 for nomenclature).

Basic write request from the engine.

Convert universal record format to database format.

Create a record in the client's Intermediate Datastore, containing for each field that could be truncated, a copy of the pre-converted field data (true value) and a copy of the field data after conversion and possible truncation (converted value).

Write the record in database format to the database.

Basic read request from engine

Read record from database

Read corresponding record from Intermediate Datastore.
Convert record to universal record format

For each field that could be truncated, compare the value returned from the application with the converted value stored in the client's Intermediate Datastore. If the values match, replace the field in the universal record with true value from the client's Intermediate Datastore.

Pass the universal record back to the engine.

This process prevents the loss of data through the propagation of truncated data during the synchronisation process.

Fig. 11 illustrates use of the Intermediate Datastore in more detail. The universal record structure Intermediate Datastore is represented on the left and the application-specific record structure is represented on the right. The client is responsible for translating the common fields in both records from one format to the other. Fig. 11 shows the universal record, having a unique id (UID) and fields Y, Z, and A – G. The application record has a corresponding UID and fields A – H. Both records have in common the UID and common fields A – G.

Fields A – D and F have the same representation in both records, so the client needs to do no processing other than copy one to the other.

Field E is represented in the application datastore in a different way to the universal record but has a reversible transformation (R) between the two representations. All the client needs to do is perform this transformation in the appropriate direction when requested to read or write a record by the engine. An example of this might be a priority field, represented in the universal contact record as values 1,2 or 3 and is represented in the application's datastore as "high", "medium" or "low".

Field G has a one-way transformation (T) from G to G". In this case universal representation and the application specific representation need to be stored in the client's Intermediate Datastore during a write operation to the application. This information is used during a read operation to determine whether field G" has been changed by the user. If it has not, The client's Intermediate Datastore's copy G is used to construct the universal record using its correct value (i.e. before transformation T has been applied). This process enables individual clients 11 to transform the data to a more suitable format whilst ensuring that any unedited transformed data is not propagated to other clients.

When synchronising records between a number of datastores the synchronisation engine ensures that, at the end of the process, each datastore contains a representation of all of the records, for example, each datastore would contain n records.

There are a number of scenarios in which it would be beneficial for one or more of these datastores to contain only a subset of the entire record set. These include, but are not limited to:

a. Limited storage capability. The datastore does not have the capacity to store all of the records. This can typically occur with devices such as mobile phones.

b. Data manageability. A large set of records may degrade the performance or usability of the application that manages them.

c. Minimising synchronisation time. Interaction with a data store may be time-consuming and so reducing the quantity of data can improve the speed of synchronisation.

d. Relevant data. A Personal Digital Assistant (PDA) user may only want records associated with their next business trip.

In this specification, the ability to synchronise a subset of records is termed Subset Personalisation. With regard to multipoint synchronisation, the engine is responsible for the actual synchronisation functionality such as conflict resolution and duplicate handling. A client that supports Subset Personalisation is responsible for managing the subset. The following describes how such a client can be implemented.

A requirement is for there to be a database containing all records in the synchronisation set. This database needs to be accessible at every synchronisation session. It also needs to contain records that support all universal field data. The master database satisfies these requirements.

The client 11 uses a Personalisation Table and a Current UID List. The Current UID List contains a list of the UIDs of all of the records in the native database. The list is generated at the start of a synchronisation session and is discarded at the end of the session. The Personalisation Table contains three columns - Master UID, Native UID and Updated. The table has a row for every entry in the Personalisation subset. The Master UID value is the UID of the record in the Master database. The Native UID value is the UID of the corresponding record in the native database. The Update value is a flag denoting whether an entry has changed. The table is persistent between sync. Sessions. Fig. 12 illustrates these components.

Subset Personalisation is achieved during synchronisation as follows. At the start of the process, the client initialises each value in the Updated column of the Personalisation Table to FALSE. It extracts a list of the UIDs and CIs of all of the records in the native database. Using these values, it updates the Change Status column in the sync table accordingly. It also creates the Current UID List from all of the UIDs in the native database.

For every new record that it finds, the client adds a new row to the Personalisation Table, setting the Native UID value to that of the new record and setting the Updated flag to FALSE. The Master UID value is set to NULL. During the synchronisation process, the engine may instruct the client to Delete, Edit or Create records. The client processes these instructions as follows:

Delete Record

The client uses the supplied Native UID to find the entry in the Personalisation Table and removes that row.

Edit Record

The client uses the supplied Native UID to find the entry in the Personalisation Table and sets the Update value of the entry to TRUE.

Create Record.

The client does nothing.

When all changes have been applied to all of the clients, the engine instructs each client to personalise its data set (if supported). When the client detects new records, it added new rows to the Personalisation Table with the Master UID set to NULL (since the Master UID is obviously unknown at that time). At this stage, as part of the synchronisation process, the UIDs of the new records will have been inserted at

the appropriate positions in the Sync Table. The client now searches the Personalisation Table for these entries. For each one that it finds, it:-

a. reads the Native UID of the entry

b. uses the Native UID to search the Sync Table to determine where the entry was applied,

c. retrieves the Master UID for that Sync Table entry, and

d. applies the Master UID to the Personalisation Table row.

The set of records that make up the Personalisation Subset can be specified in a number of ways. These include, but are not limited to:

a. User selection. The user is provided with a list of all of the records and can select those records that are to make up the subset.

b. Rule based selection. The user sets some criteria that a record must satisfy for inclusion in the subset e.g. Country = USA.

A key difference between these two methods is that, in the former, the Personalisation Set can vary during the process due to deletions in any client or new records in the native database. However in the latter method, the Personalisation Set can vary during the process as a result of deletions, edits and new records from any client. At this stage, the master database contains all of the records with the latest data. A client using the rule based method now needs to update its Personalisation Table entries. For each entry in the Personalisation Table, the client:

a. reads the Master UID,

b. retrieves the associated record from the Master database, and

c. checks whether the rule criteria are satisfied. If not, that entry is removed from the Personalisation Table.

The client now reads each record in the master database. For each record, it checks whether the rule criteria are satisfied. If they are, and the entry is not currently in the set, the client creates a new entry in the Personalisation Table. The new entry has the Master UID value applied, the Updated field set to FALSE and the Native UID value set to NULL. The Native UID value will be set when the record is created in the native database.

At this point,

a. the Personalisation Table is now fully populated with the correct data and contains an entry for each record in the subset.

b. the Master database contains all of the records with the latest data.

c. the Current UID List contains a list of the UIDs of all of the records currently in the native database.

The client now applies the actions (Delete, Edit, Create) needed for the Personalisation Subset.

Delete

If a Native UID in the Current UID List does not have a corresponding entry in the Personalisation Table then that record is deleted in the Native database.

Create

If a Native UID value in the Personalisation Table is NULL then a new record is required in the Native database. The client 11 uses the Master UID entry to retrieve the correct record from the Master database and creates a corresponding record in the native database. The UID of the newly created record is stored in the Native UID column.

Edit

If a Native UID in the Current UID List has a corresponding entry in the Personalisation Table and the Update value is TRUE then that record needs to be edited in the native database. The client uses the Master UID entry to retrieve the correct record from the Master database and updates the corresponding record in the native database.

When all actions have been applied, the subset of records are synchronised and the synchronisation process is complete.

It will be appreciated that the invention achieves the following advantages.

Ability to control the records from the overall available set of records which are required to be synchronised to any given data store, by using either explicit selection or rules based selection.

Enabling datastores of limited storage ability to be synchronized with large data stores.

Improving data managability.

Reducing synchronisation time.

Allowing the user to synchronize only relevant data.

Having a (well thought out) defined universal record format at the outset enables a developer to provide all the mappings and transformations within the engine during the coding stage and only needs to require the user to map fields during the setup of the synchronisation in extreme cases. This reduces the risk of user error and confusion. It also simplifies the process of user configuration of the synchronisation process.

Furthermore, by ensuring that the engine stores its data in the master datastore in the universal record format, no internal transformations are required and all comparisons required to do the synchronisation are independent of the applications being synchronised. This enables the incremental development and addition of new clients without requiring any change to the engine.

Exposing the universal record format to the user through the user interface has a number of additional benefits; namely:

- The user can get a view of his synchronised data in a common format.
- The user can see all data that is partaking in the synchronisation (viewing it through the interface of one of the Application may not expose all the information partaking in the synchronisation.)
- When displaying conflicts in records from two or more applications, the user is able to see the conflicts in a common format.

Thus, the invention enables the incremental adding of clients to a multi-point synchronisation process without the need to:

- test every other synchronisation client,
- be aware of any other record format other than the universal record format, and
- to amend the core synchronisation product.

The invention is not limited to the embodiments described but may be varied in construction and detail.

Claims

1. A synchronisation system comprising a synchronisation engine comprising means for communication with a plurality of client databases and for updating the client databases with current data received by a client database, characterised in that,

the synchronisation engine (10) comprises means for performing a synchronisation session in which all client databases (DBA-DBD) are updated with current data.

2. A synchronisation system as claimed in claim 1, wherein said engine comprises means for maintaining a synchronisation table (20) storing an identifier and a change indicator for each record of each client database, for comparing the stored change indicators with those read from the client databases during a synchronisation session, and for maintaining the table in a persistent manner between synchronisation sessions.

3. A synchronisation system as claimed in claim 2, wherein the system comprises a client interface (11) associated with each client database, each client interface comprising means for providing a universal format for transfer of data to and from the engine independently of the client database format.

4. A synchronisation system as claimed in any preceding claim, wherein the synchronisation engine (10) comprises means for prompting user input of instructions if a conflict arises in which a record has changed in more than one client database.

5. A synchronisation system as claimed in claim 4, wherein the engine (10) comprises means for determining that there is a conflict if there are differences between records at the field level.

6. A synchronisation system as claimed in claims 4 or 5, wherein the engine (10) comprises means for outputting representations of the conflicting fields in a universal format.

7. A synchronisation system as claimed in any preceding claim, wherein the engine (10) comprises means for adding or deleting databases as clients by adding or deleting corresponding datasets such as columns in the synchronisation table.

8. A synchronisation system as claimed in any preceding claim, wherein the engine (10) comprises means for maintaining a master database of current data.

9. A synchronisation system as claimed in claim 8, wherein the engine comprises means for using the master database as a client database when a client database is temporarily unavailable.

10. A synchronisation system as claimed in claim 9, wherein the engine (10) comprises means for temporarily masking a dataset for a temporarily unavailable client database.

11. A synchronisation system as claimed in claim 10, wherein the engine (10) comprises means for maintaining a status indicator (5) in the synchronisation table to indicate availability status of a corresponding client database.

12. A synchronisation system as claimed in claim 11, wherein the status values include uncreated, unsynchronised, synchronised, and null.

13. A synchronisation system as claimed in any preceding claim, wherein a client interface comprises means for maintaining an intermediate database of true values for truncated values in the associated client database.

14. A synchronisation system as claimed in any preceding claim, wherein a client interface comprises means for maintaining a persistent database to allow transformation to provide a universal format to the engine.

15. A synchronisation system as claimed in any preceding claim, wherein a client interface comprises means for maintaining a personalisation table containing a master identifier, a native identifier, an updated flag for every associated record, and a current record identifier list, and the client interface comprises means for using said data to maintain a client database having a subset of the records of other client databases.

16. A method for synchronising a plurality of client databases in a single synchronisation session, the method comprising the steps of:

storing a synchronisation table of record identifiers and change indicators for records of the client databases;

receiving the values for said identifiers and change indicators from the client databases, and comparing the received values with the stored values to determine if a record has changed to determine the current value; and

updating all of the client databases with a current value of a record and updating the synchronisation table for a next synchronisation session.

17. A computer program product comprising software code for performing the method of claim 16 when executing on a digital computer.

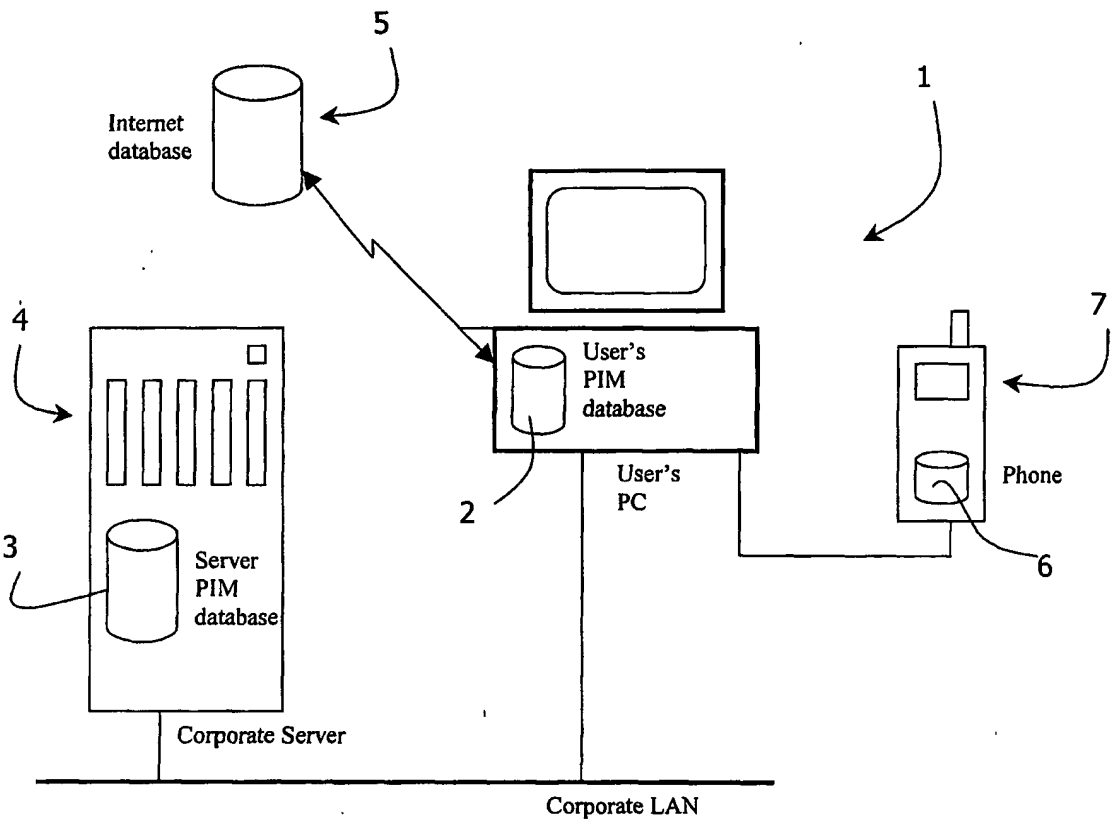

5
Internet database
1
4
User's PIM database
7
Phone
User's PC
2
3
Server PIM database
6
Corporate Server
Corporate LAN

Fig. 1

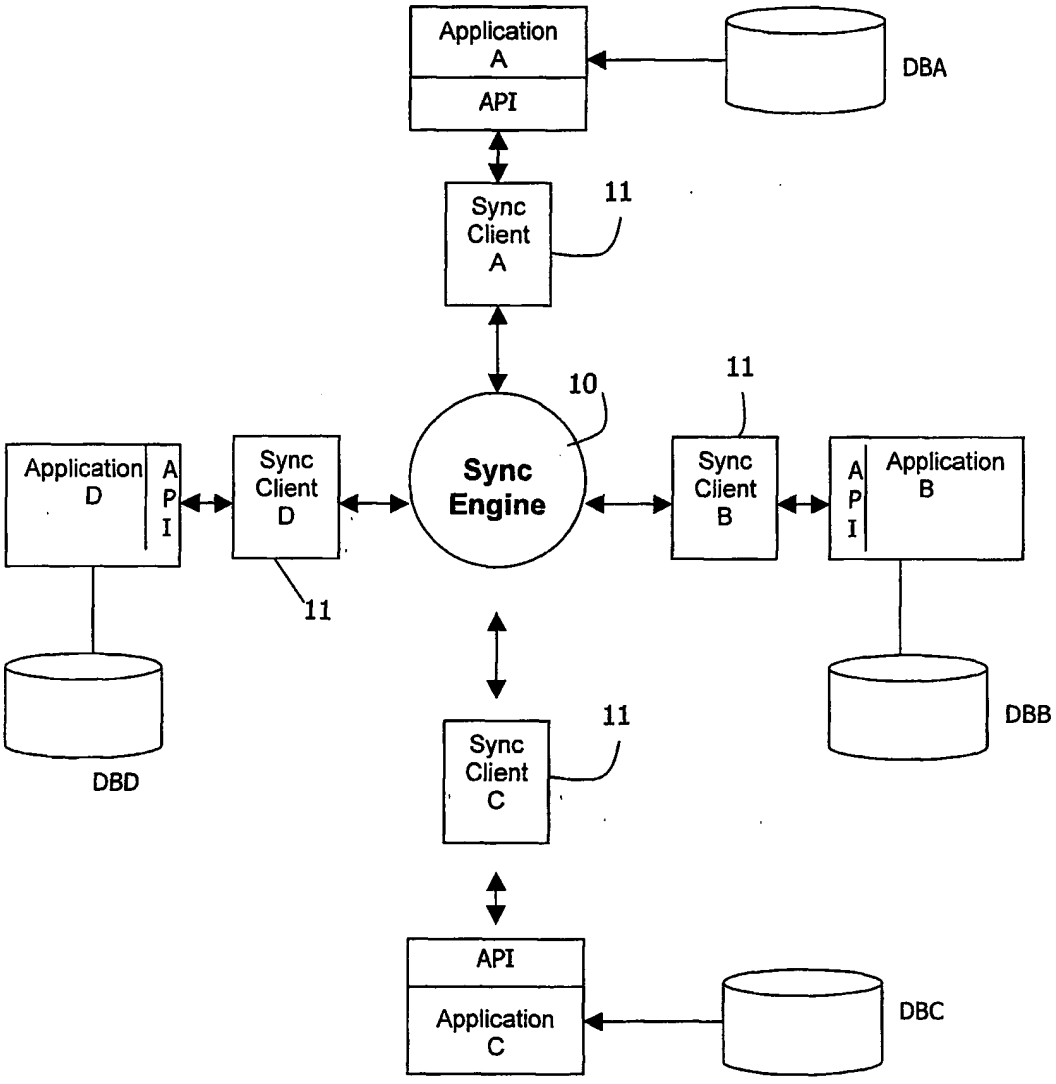

Application A
API
DBA
Sync Client A
11
10
11
Application D
API
Sync Client D
Sync Engine
Sync Client B
API
Application B
11
DBD
DBB
Sync Client C
11
API
Application C
DBC

Fig. 2

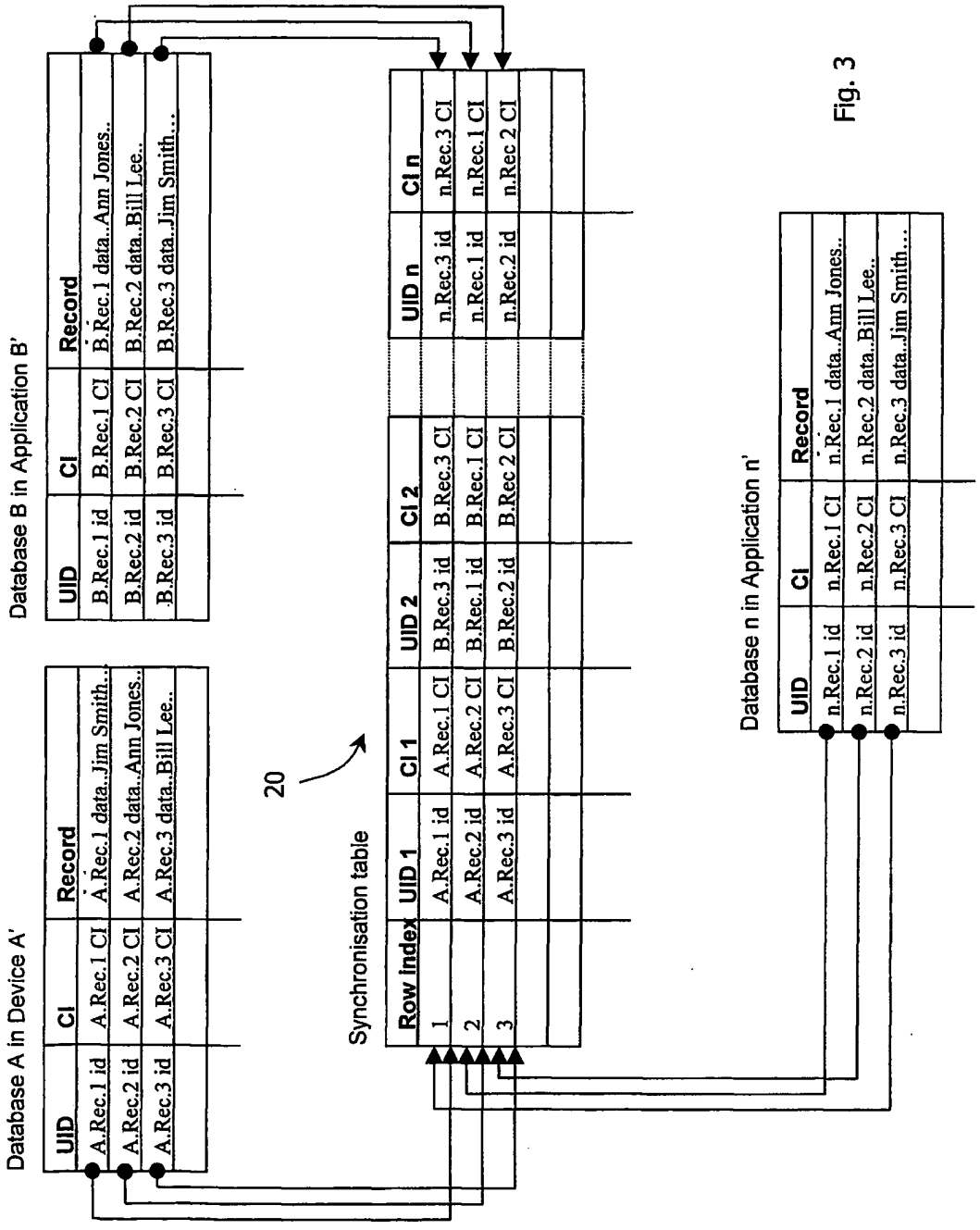

Database A in Device A'
UID
CI
Record
A.Rec.1 id
A.Rec.1 CI
A.Rec.1 data..Jim Smith..
A.Rec.2 id
A.Rec.2 CI
A.Rec.2 data..Ann Jones..
A.Rec.3 id
A.Rec.3 CI
A.Rec.3 data..Bill Lee..
Database B in Application B'
UID
CI
Record
B.Rec.1 id
B.Rec.1 CI
B.Rec.1 data..Ann Jones..
B.Rec.2 id
B.Rec.2 CI
B.Rec.2 data..Bill Lee..
B.Rec.3 id
B.Rec.3 CI
B.Rec.3 data..Jim Smith...
20
Synchronisation table
Row index
UID 1
CI 1
UID 2
CI 2
UID n
CI n
1
A.Rec.1 id
A.Rec.1 CI
B.Rec.3 id
B.Rec.3 CI
n.Rec.3 id
n.Rec.3 CI
2
A.Rec.2 id
A.Rec.2 CI
B.Rec.1 id
B.Rec.1 CI
n.Rec.1 id
n.Rec.1 CI
3
A.Rec.3 id
A.Rec.3 CI
B.Rec.2 id
B.Rec.2 CI
n.Rec.2 id
n.Rec 2 CI
Database n in Application n'
UID
CI
Record
n.Rec.1 id
n.Rec.1 CI
n.Rec.1 data..Ann Jones..
n.Rec.2 id
n.Rec.2 CI
n.Rec.2 data..Bill Lee..
n.Rec.3 id
n.Rec.3 CI
n.Rec.3 data..Jim Smith...

Fig. 3

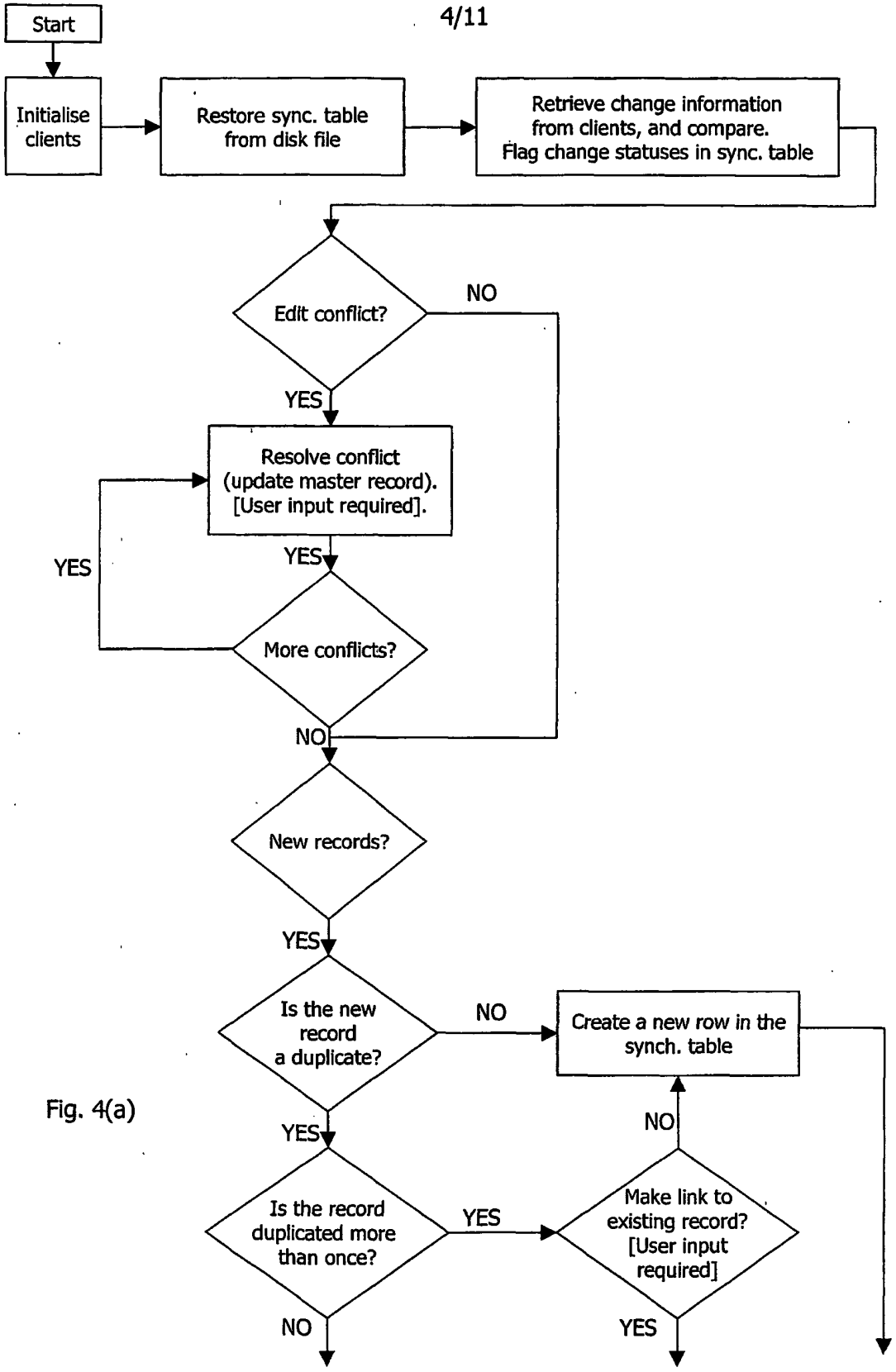

Start
Initialise clients
Restore sync. table from disk file
Retrieve change information from clients, and compare. Flag change statuses in sync. table
Edit conflict?
NO
YES
Resolve conflict (update master record). [User input required].
YES
YES
More conflicts?
NO
New records?
YES
Is the new record a duplicate?
NO
Create a new row in the synch. table
YES
NO
Is the record duplicated more than once?
YES
Make link to existing record? [User input required]
NO
YES

Fig. 4(a)

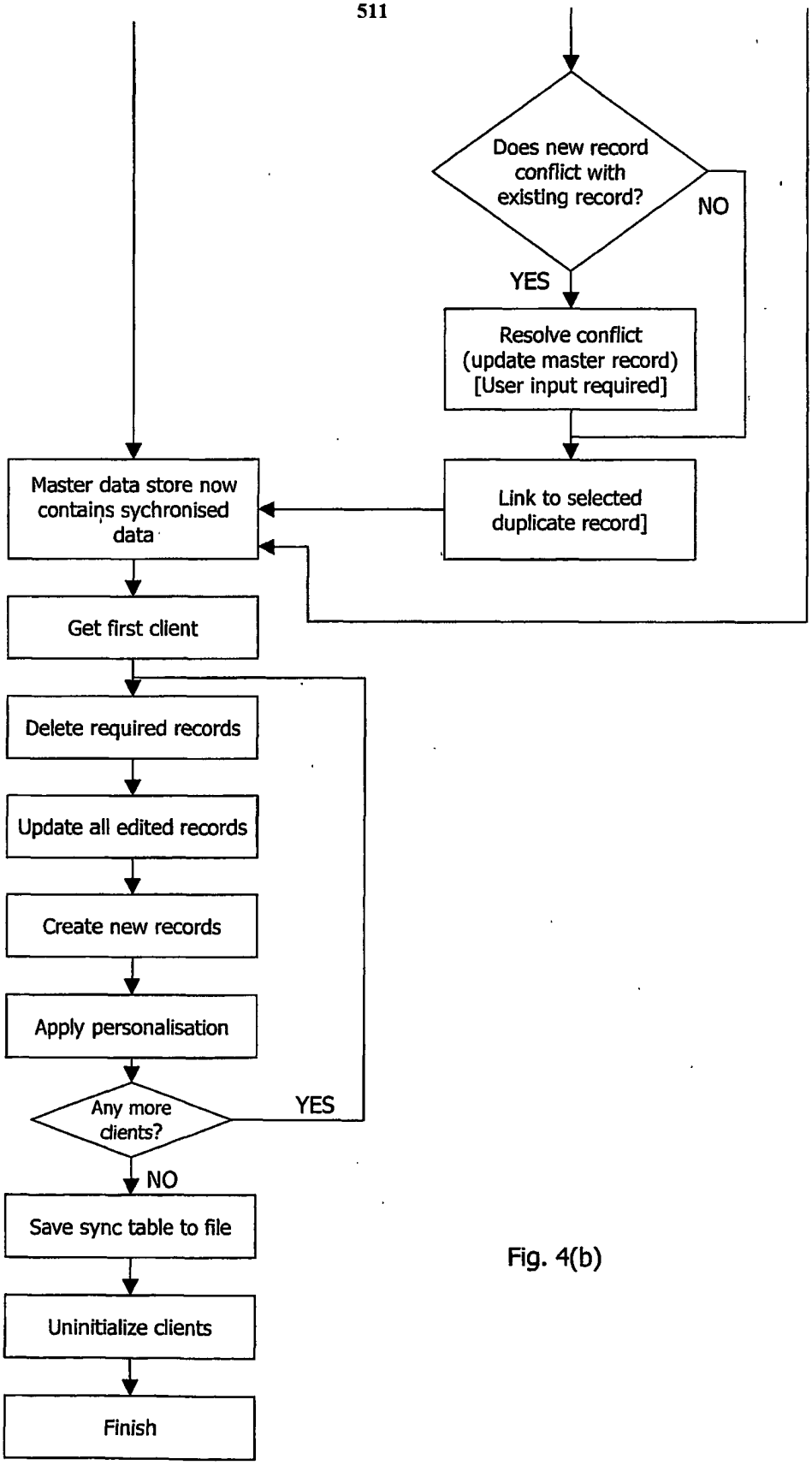

Does new record conflict with existing record?
NO
YES
Resolve conflict (update master record) [User input required]
Link to selected duplicate record]
Master data store now contains sychronised data
Get first client
Delete required records
Update all edited records
Create new records
Apply personalisation
Any more clients?
YES
NO
Save sync table to file
Uninitialize clients
Finish

Fig. 4(b)

Database A in Application A'

UID	CI	K				
123	777	12	Apples	Oranges	Pears	Bananas

Database in Application B'

UID	CI	K				
789	11:15:48	12	Apples	Oranges	Pears	Bananas

Synchronisation Table

Row index	UID A	CI A	UID B	CI B
1				
2	123	777	789	11:15:48
3				

FIG. 5: System state after last synchronisation

Database A in Application A'

UID	CI	K				
123	778	12	Plums	Oranges	Kiwis	Bananas

Database in Application B'

UID	CI	K				
789	16:45:18	12	Cherries	Oranges	Kiwis	Peaches

Synchronisation Table

Row index	UID A	CI A	UID B	CI B
1				
2	123	778	789	16:45:18
3				

FIG. 6: System state prior to next synchronisation

	Source	K	Fld1	Fld2	Fld3	Fld4
	Merged Record	12	*Cherries*	*Bananas*	Kiwis	Oranges
▶	Database A	12	*Plums*	*Bananas*	Kiwis	Oranges
▶	Database B	12	*Cherries*	*Peaches*	Kiwis	Oranges

Fig. 7: Conflict Resolution dialog

Database A in Application A'

UID	CI	K	Fruit1	Fruit2	Fruit3	Country
123	778	12	Plums	Oranges	Kiwis	England

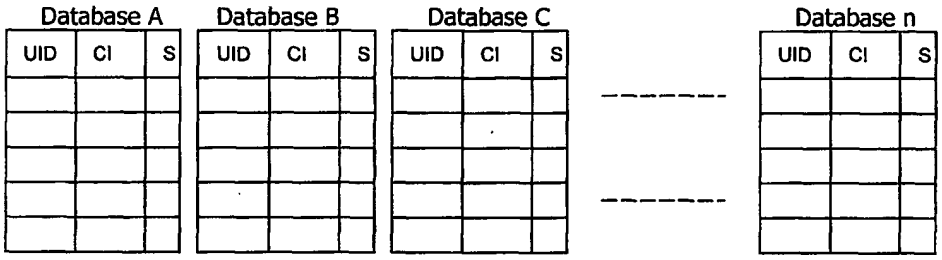
Database B in Application B'

UID	CI	K	Fruit1	Fruit2	Fruit3	Colour
789	16:45:18	12	Plums	Oranges	Kiwis	Blue

Synchronisation Table

Row index	UID A	CI A	UID B	CI B
1				
2	123	778	789	16:45:18
3				

Fig. 8: Non intersecting sets of changed fields



Key: UID Universal Identifier
CI Change Indicator
S SyncStatus

Fig. 9: Synchronisation Table for Disconnectable Clients

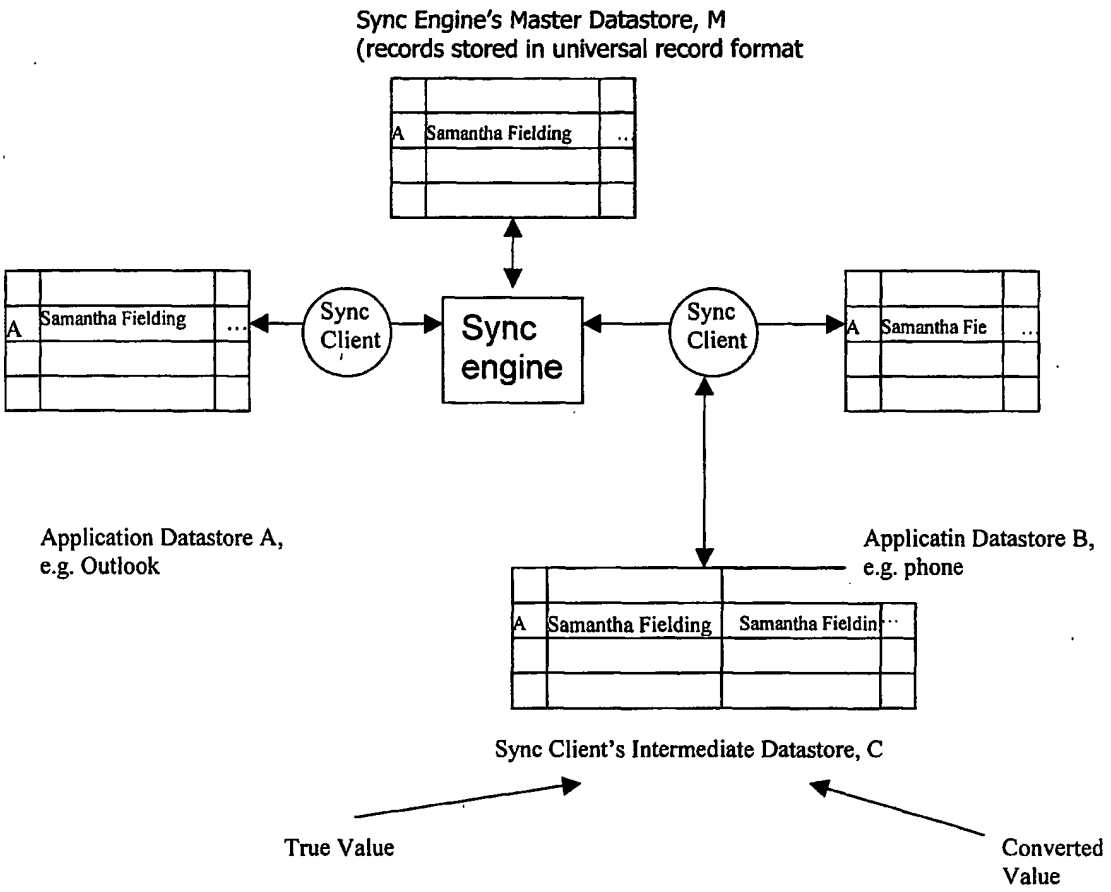

Sync Engine's Master Datastore, M
(records stored in universal record format
A Samantha Fielding ...
A Samantha Fielding ...
Sync Client
Sync engine
Sync Client
A Samantha Fie ...
Application Datastore A,
e.g. Outlook
Applicatin Datastore B,
e.g. phone
A Samantha Fielding Samantha Fieldin...
Sync Client's Intermediate Datastore, C
True Value
Converted Value

Fig. 10

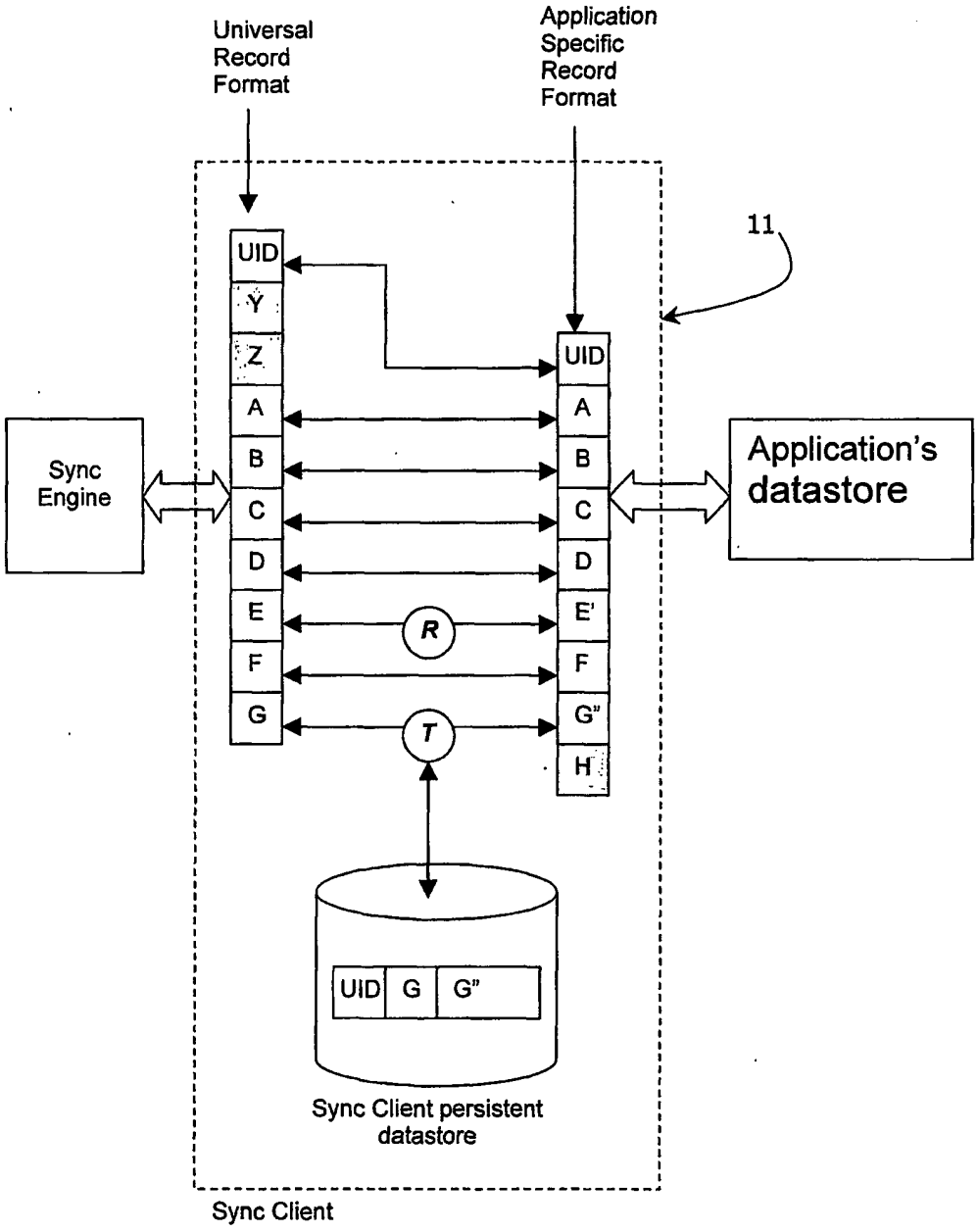


Fig. 11: Logical view of a Sync Client's transformation operation

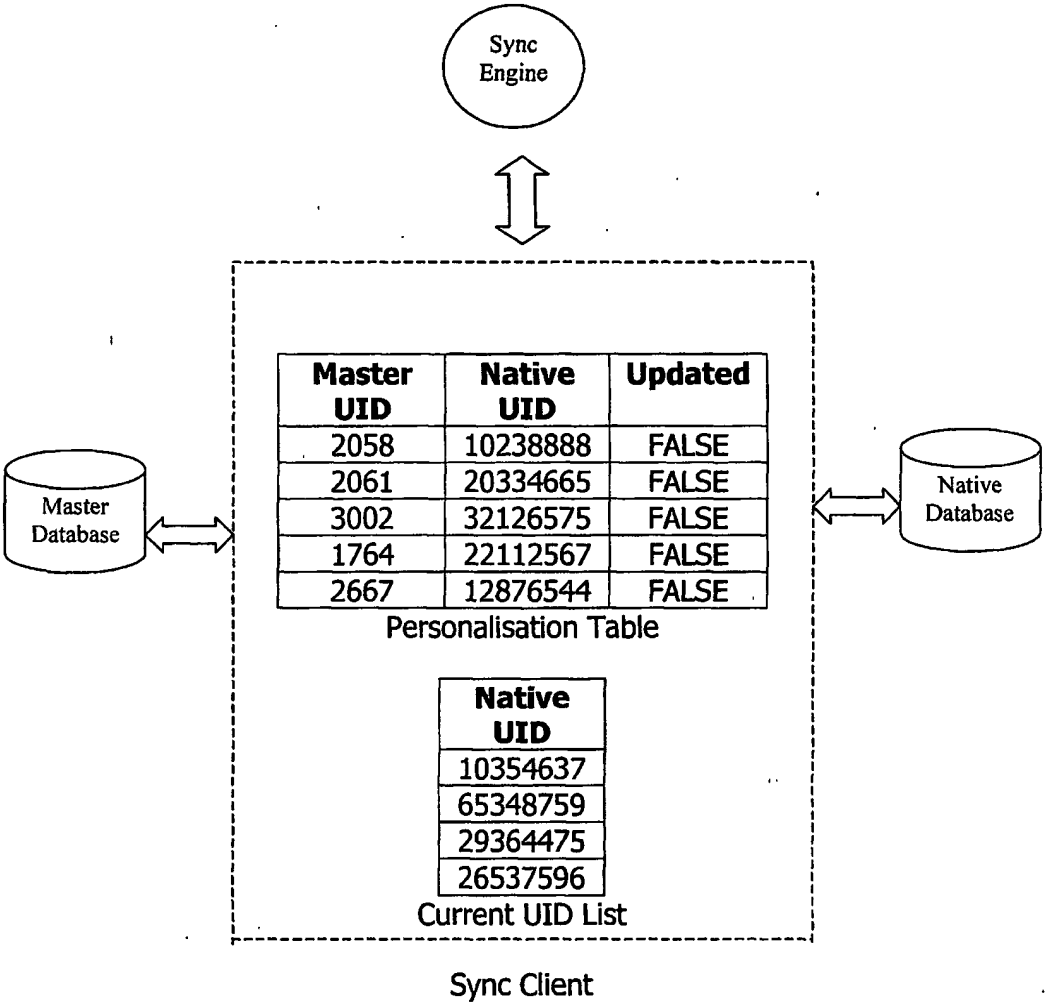


Figure 12: Subset Personalisation components

INTERNATIONAL SEARCH REPORT

Inter I A[redacted]on No
PCT/EP 01/05869

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, IBM-TDB, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 884 325 A (BODGE ANDREW ET AL) 16 March 1999 (1999-03-16) abstract column 1, line 50 -column 2, line 46 column 4, line 13 -column 4, line 63	1,8,16, 17
A		2-7,9-15

X	US 5 999 947 A (DEVINE JOHNATHAN ET AL) 7 December 1999 (1999-12-07) abstract column 5, line 8 -column 6, line 40 column 8, line 9 -column 8, line 64	1,14,16, 17
A		2-13,15

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 August 2001	06/09/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Boyadzhiev, Y

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 684 990 A (BOOTHBY DAVID J) 4 November 1997 (1997-11-04) column 3, line 15 -column 4, line 2 column 4, line 21 -column 4, line 51 column 6, line 8 -column 6, line 27	1,4,16, 17
A	---	2,3,5-15
A	US 5 974 238 A (CHASE JR CHARLIE DAVID) 26 October 1999 (1999-10-26) abstract column 3, line 26 -column 3, line 52	1,16,17

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter[illegible]al Application No
PCT/EP 01/05869

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5884325 A	16-03-1999	NONE	
US 5999947 A	07-12-1999	AU 7598498 A	30-12-1998
		EP 1016004 A	05-07-2000
		WO 9854662 A	03-12-1998
US 5684990 A	04-11-1997	AU 5019496 A	31-07-1996
		WO 9621898 A	18-07-1996
US 5974238 A	26-10-1999	NONE	